Estimating the distribution in an EDA

S. Shakya, J. McCall, D. F. Brown
School of Computing, The Robert Gordon University, Aberdeen, UK
E-mail: {ss, jm, db}@comp.rgu.ac.uk

Abstract

This paper presents an extension to our work on estimating the probability distribution by using a Markov Random Field (MRF) model in an Estimation of Distribution Algorithm (EDA) [1]. We propose a method that directly samples a MRF model to generate new population. We also present a new EDA, called the Distribution Estimation Using MRF with direct sampling (DEUM$_d$), that uses this method, and iteratively refines the probability distribution to generate better solutions. Our experiments show that the direct sampling of a MRF model as estimation of distribution provides a significant advantage over other techniques on problems where a univariate EDA is typically used.

1 Introduction

Estimation of Distribution Algorithms (EDAs) [2][3], also known as Probabilistic Model Building Genetic Algorithms (PMBGAs) [4], are a well-established topic in the field of evolutionary algorithms. EDAs are motivated by the idea of identifying important patterns or building blocks [4] from the population of promising solutions. A model of the probability distribution is used to preserve those patterns and is explicitly sampled to generate a child population. EDAs are classified as univariate, bivariate or multivariate according to the type of interaction between allele values that is allowed in the model of the probability distribution (see [3][4]).

An EDA regards a solution (chromosome) as a set of random variables (the alleles), each taking a particular value from a set of possible values. In particular, we represent a solution as $x = \{x_1, x_2, \ldots, x_n\}$ where each x_i is the value taken by the i-th random variable. Univariate EDAs do not consider dependencies between variables, i.e., they only model building blocks of order one. In this case, the joint probability distribution, $p(x)$, is simply the product of the univariate marginal probabilities of all variables in a solution x:

$$p(x) = \prod_{i=1}^{n} p(x_i) \tag{1}$$

Here, $p(x_i)$ is the marginal probability of the i-th variable having the value x_i. Population Based Incremental Learning (PBIL), the Univariate Marginal Distribution Algorithm (UMDA), and the Compact Genetic Algorithm (cGA) all use a univariate model of the probability distribution (see [3][4][5]).

In our recent work [1], we propose a different model of probability distribution for EDAs known as Markov Random Field (MRF) model [6]. As in PBIL, the algorithm proposed there, known as *Distribution Estimation Using MRFs* (DEUM), maintains a probability vector, however, uses a univariate MRF model to update it [1].

In this paper, we refine our use of MRF models for the estimation of distribution. We describe an updated version of DEUM called the *Distribution Estimation Using MRFs with direct sampling* (DEUM$_d$). DEUM$_d$ does not maintain a probability vector. Instead, it directly samples the MRF model to generate new population. The workflow of DEUM$_d$ is more similar to that of UMDA than PBIL. In UMDA, the marginal frequencies are directly sampled to generate successive populations. In DEUM$_d$, we replace these marginal frequencies with a MRF model that is also built from a selected subset of the population. This MRF model gives a maximum likelihood estimation of the optimal solution based on the selected set, and it is sampled to generate a successive population. The result of this, as we will show, is a significant improvement in learning on well-known univariate EDA problems.

The rest of the paper is constructed as follows. Section 2 introduces our univariate MRF model, and shows how we determine the model from a population. Section 3 describes the operation of DEUM$_d$ in detail, and Section 4 gives the results of several experiments that compare DEUM$_d$ with other univariate EDAs. Finally, Section 5 summarises and outlines further work.

2 A univariate MRF model of fitness

In [7], MRF theory was used to provide a formulation of the joint probability distribution that relates solution fitness to an energy function calculated from the values of the solution variables. To be precise:

$$p(x) = \frac{f(x)}{\sum_y f(y)} = \frac{e^{-U(x)/T}}{\sum_y e^{-U(y)/T}} \tag{2}$$

from which we can derive an equation for each solution x (see [7]):

$$-\ln(f(x)) = U(x) \tag{3}$$

Here, $f(x)$ is the fitness of an individual. $U(x)$ is an energy function derived from allele values and, T is a *temperature coefficient*. The summations are over all possible solutions y. $U(x)$ gives the full specification of the joint probability distribution, so it can be regarded as a probabilistic model of the fitness function. In particular, minimising $U(x)$ is equivalent to maximising $f(x)$. In general, the form of the energy function will involve interactions between the variables x_i. In DEUM$_d$, however, we use a simple form that assumes no such interactions. Instead each variable provides a contribution $\alpha_i x_i$ to the overall energy. From the above, we can derive an equation for each solution:

$$-\ln(f(x)) = \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n \tag{4}$$

We refer to this as the *univariate MRF model*. The real-valued α_i are called the MRF parameters, and completely determine the probability distribution.

Each solution in a given population provides an equation satisfying the model. Selecting N promising solutions from a population therefore allows us to estimate the distribution by solving the system of equations:

$$A\alpha^T = F \tag{5}$$

Here, A is the $N \times n$-dimensional matrix of allele values in the selected set, α is the vector of MRF parameters $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, and F is the N-dimensional vector containing $-\ln(f(x))$ of the selected set of solutions x. Solving this system of linear equations, we get the set of MRF parameters α. Depending on the relationship between N and n, the system will be under-, over-, or precisely-specified. A standard fitting algorithm can be used to give a maximum likelihood estimation of the α_i. The α_i can then be used to provide a direct estimate of the probability of the value of x_i. For mathematical reasons, we use $\{-1, 1\}$ as the values of x_i in our model, rather than $\{0, 1\}$. This ensures arithmetical symmetry between the possible allele values. However, the following analysis generalises to any choice of distinct values.

Fixing the value of a particular allele divides the set Ω of all chromosomes into two disjoint sets, which we denote by A and B. More precisely, $A = \{x \in \Omega : x_i = 1\}$ and $B = \{x \in \Omega : x_i = -1\}$. We denote the probability that the allele value in position i is equal to 1 by $p(x_i = 1)$. Clearly, the probability that the allele value in position i is equal to -1 is $1 - p(x_i = 1)$. From (2), we obtain:

$$p(x_i = 1) = \sum_{x \in A} p(x) = \sum_{x \in A} \frac{e^{-U(x)/T}}{Z} \tag{6}$$

Here, $Z = \sum_y e^{-U(y)/T}$ is a (very large) normalising constant. Substituting for $U(x)$ from (4), and noting that $x_i = 1$ for all $x \in A$, we obtain:

$$p(x_i = 1) = e^{-\alpha_i/T} \frac{K}{Z} \tag{7}$$

where K is a large constant representing the sum over all chromosomes in A of contributions from alleles in positions other than i.

Similarly, summing over B we obtain the probability that the allele value in position i is equal to -1:

$$p(x_i = -1) = 1 - p(x_i = 1) = e^{\alpha_i/T} \frac{K}{Z} \tag{8}$$

Here, K is the same constant as in (7), because the chromosomes in A and B agree pairwise at allele positions other than i. Combining (7) and (8), the constants K and Z drop out, and we get the following expression as an estimate of the marginal probability for $x_i = 1$:

$$p(x_i = 1) = \frac{1}{1 + e^{\beta \alpha_i}} \tag{9}$$

where, $\beta = 2/T$.

Note that, as $T \to 0$, the value of β increases, and the value of $p(x_i = 1)$ tends to an extreme depending on the sign of α_i. If $\alpha_i > 0$, then $p(x_i = 1) \to 0$ as $T \to 0$. Conversely, if $\alpha_i < 0$, then $p(x_i = 1) \to 1$ as $T \to 0$. If $\alpha_i = 0$, then $p(x_i = 1) = 0.5$ regardless of the value of T. Therefore, the α_i are indicators of whether the allele value at the position i should be 1 or -1. This indication becomes stronger as the temperature is cooled towards zero.

This forms the basis for our estimation of distribution technique, which combines the univariate MRF model with a cooling scheme. We reduce T, i.e., increase β, as the population evolves, so the model becomes more exploitative rather than explorative as the evolution progresses.

3 DEUM$_d$: Distribution Estimation Using MRFs with direct sampling

The five step procedure of the algorithm for DEUM$_d$ are as follows:

1. Generate an initial population, P, of size M with uniform distribution.

2. Select the N fittest solutions from P, where $N \leq M$.

3. Calculate the MRF parameters $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ by applying the univariate MRF model to the selected solutions and solving the system of linear equations.

4. Generate M new solutions using the following distribution:

$$p(x) = \prod_{i=1}^{n} p(x_i)$$

where, $p(x_i = 1) = \frac{1}{1+e^{\beta\alpha_i}}$ and $p(x_i = -1) = \frac{1}{1+e^{-\beta\alpha_i}}$. Here, β is defined as $\beta = g\tau$ where, g is the number of the current iteration and $\tau > 0$ is a *cooling rate* parameter chosen by the user.
5. Replace P by the new population, and go to Step 2 until the termination criterion is satisfied.

DEUM$_d$ uses the singular value decomposition (SVD) [8] technique to solve the system of linear equations. SVD proves to be the most stable technique, and can solve (in the sense of giving a useful numerical answer) systems of linear equations that are either under- or over-specified [8].

As described in Section 2, β has a direct effect on the convergence speed of DEUM$_d$. As the number of iterations (g) grows, the marginal probability ($p(x_i)$) gradually cools down to either 0 or 1. However, depending upon the type of problem, different cooling rate may be required. In particular, there is a trade-off between convergence speed of the algorithm and the exploration of the search space. Therefore, the cooling rate parameter, τ, has been introduced. τ gives the user explicit control over the convergence speed of DEUM$_d$. Decreasing τ slows the cooling, resulting in better exploration of the search space. However, it also slows the convergence of the algorithm. Increasing τ, on the other hand, makes the algorithm converge faster. However, the exploration of the search space will be reduced.

4 Experiments

In this section, we compare DEUM$_d$ with other univariate EDAs including DEUM and a GA, on two different problems. In order to compare best with best, we empirically determined the parameters for DEUM$_d$. For the rest of the algorithms, we used parameter settings from the literature or empirically determined parameters, depending on which proved best for particular problems. The performance of each algorithm was measured in terms of the number of fitness evaluation taken to find the optimal solution.

4.1 Onemax Problem

The Onemax problem [2] is a simple linear problem decomposable into building blocks of order one, and therefore is an ideal problem for univariate EDAs. It has been shown that UMDA works very well on this problem [2]. We compare the performance of DEUM$_d$ against a simple GA with uniform crossover (GA (uniform)), UMDA, and DEUM. 100 runs of each algorithm were executed for a series of Onemax problems with chromosomes ranging in size between 30 and 180. The number of fitness evaluations taken to find the optimal solution was recorded for each run. Uniform crossover with exchange probability of 0.5 was used for GA (uniform), crossover was applied all the time and mutation was not applied. Population size M ranged from 40 to 100 for GA (uniform), 50 to 170 UMDA, $1.5n$ for DEUM and was fixed at 40 for DEUM$_d$. λ for DEUM was from 0.5 to 0.6 and τ for DEUM$_d$ was from 5 to 4.

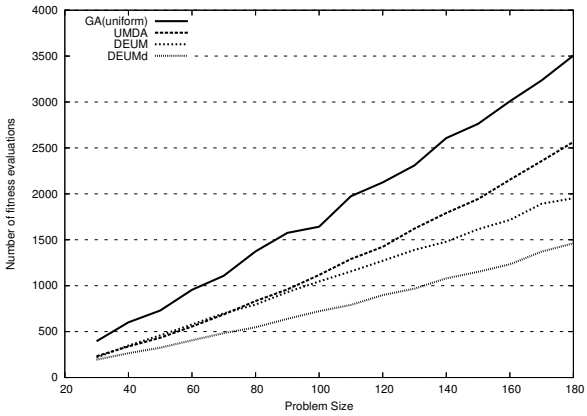

Fig. 1. Average number of fitness evaluations for 30 to 180 sized onemax problem where the population size was 40 to 100 for GA (uniform), 50 to 170 for UMDA, $1.5n$ for DEUM and 40 for DEUM$_d$.

Truncation selection was used where selection size N was $0.5M$ for GA (uniform), $0.3M$ for UMDA, $0.85M$ for DEUM and a fixed size of 10 was used for DEUM$_d$. No elitism was used and new populations were generated with complete replacement. Fig. 1 shows the average number of fitness evaluations for each algorithm over the range of onemax problems.

The success ratio of converging to the optimum was 93.5% for DEUM, 96% for DEUM$_d$, 98% for UMDA and 100% for GA (uniform). As we can see from Fig. 1, UMDA has an expected performance, better than that of GA (uniform) [2]. However, DEUM$_d$ performs better than all of the other algorithms, for all problem instances independent of their size.

4.2 Schaffer f6 function optimization

The Schaffer f6 function, described in [9] has been frequently used to evaluate the performance of GAs. An interesting feature of this function is that it has many local optima, but a single global optimal solution. So a hill-climbing algorithm will rapidly become trapped in one of the local optima. A simplified version of it is presented below:

$$f(x) = 1 + \left(\frac{\cos(x)}{1 + 0.001x^2}\right)$$

where $-300 \leq x \geq 300$.

The optimal solution is $f(x) = 2$ when $x = 0$. We performed experiments with a 20-bit Gray code representation of the f6 function.

Each algorithm with fixed parameter settings was run for total of 1000 runs. For each run the number of evaluations taken to find the optimum was recorded. For GA(uniform), the population size was 300, and truncation selection with a selection size $N = 0.5M$ was used. Crossover was applied all the time, mutation was set to 0.01 and 50% elitism was used. For PBIL, DEUM and DEUM$_d$, the population size was 500 and the selection size, N was 2. The learning rate, λ, for both PBIL and DEUM was 0.1 and the cooling rate, τ, for DEUM$_d$ was 1.5. Mutation shift was not applied in PBIL.

The experimental results are shown in Fig. 2, where the Run Length Distribution (RLD) [10] is plotted for each of the compared algorithms. We can see that, with DEUM$_d$, 80% of runs found the optimum within 4500 function evaluations, compared with 9000 function evaluations for PBIL. The success rate for finding a solution for DEUM$_d$ was 92% compared to 94% of DEUM, however the number of function evaluations needed to find the solution for DEUM$_d$ was significantly less than that of other algorithms.

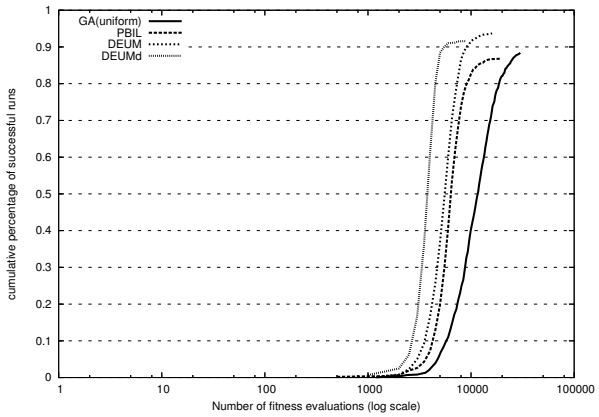

Fig. 2. Experimental results in the form of RLD showing, for each algorithm running on 20-bit Schaffer f6 function, the cumulative percentage of successful runs that terminated within a certain number of function evaluations.

5 Conclusions

In this paper, we have presented DEUM$_d$ as a novel EDA, which uses MRF modelling of fitness to estimate the probability distribution. The motivation behind DEUM$_d$ is to use a direct sampling of a MRF model to generate a new population in order to improve evolution. Our experiments shows that, for univariate problems, the use of MRF parameters instead of marginal probabilities does provide a better estimation of the distribution. This leads to better performance in terms of the number of function evaluations required for convergence to the global optimum. There are some penalties though. Calculating the MRF parameters is computationally more expensive than calculating marginal distributions, and so DEUM$_d$ will be particularly appropriate for problems where there is a positive trade-off in reducing the number of fitness evaluations.

A promising line of research in this area is to develop MRF models for bivariate and multivariate EDAs, where the extra computational costs are more likely to be compensated by a reduction in the number of fitness evaluations required to solve higher-order problems.

6 References

[1] Shakya, S., McCall, J., Brown D. (2004). Updating the probability vector using MRF technique for a Univariate EDA . In proceedings of STAIRS 2004, IOS press, pp. 15–25.

[2] Mühlenbein, H., Paass, G. (1996). From recombination of genes to the estimation of distributions I. Binary parameters. Voigt, H.-M et. El.(eds.) PPSN IV, LNCS 1141, Springer, pp. 178–187.

[3] Larrañaga P., and Lozano J. A. (2001) Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, 2001.

[4] Pelikan, M., (2002). Bayesian optimization algorithm: From single level to hierarchy. Ph.D. thesis, University of Illinois at Urbana -Champaign, Urbana, IL.

[5] Baluja, S. (1994). Population based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMUCS94163, Pittsburgh, PA.

[6] Li, S. Z. (1995). Markov random field modeling in computer vision. ISBN:4-431-70145-1, Springer-Verlag,1995.

[7] Brown D.F., Garmendia-Doval, A.B., McCall, J.A.W. (2001). Markov Random Field Modelling of Royal Road Genetic Algorithms. Evolution Artificielle 2001, LNCS 2310, pp. 65–78, Springer Verlag 2002.

[8] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P (1993). Numerical Recipes in C: The Art of Scientific Computing. Cambridge University Press, 2nd edition, 1993

[9] Lawrence Davis (1991), editor. Handbook of Genetic Algorithms. Van Nostrand Reinhold, 1991.

[10] Hoos, H. H. and Stutzle, T. (1999). Towards a characterisation of the behaviour of stochastic local search algorithms for SAT. Artificial Intelligence, 112(1-2):213.232, 1999.